58同城跨平台高性能高可用中间层服务架构设计分享

58同城 陈春

2012-09-10

关于我

姓名：陈春

58同城 - 技术中心 - 架构部

在58同城主要负责基础架构，跨平台分布式服务，通用组件，基础支撑框架的设计与研发相关工作. 有幸参与了58同城多个重大核心系统的设计与研发工作

sina微博：@58陈春

关于这次分享

主要介绍如何塔建一个跨平台，分布式，高性能，高可用的中间层服务，以及58.com开源的中间层服务框架(容器)－－ Gaea(盖亚)

目录:

1. 设计异构平台高可用，高性能的中间层服务的挑战
2. 常见的解决方案
3. 58.com的解决方案
4. Gaea的设计和实现细节

设计一个异构平台中间层服务有哪些挑战？

异构平台 => 如何跨平台？
如何通讯，采用哪种通讯模型？
采用什么协议？
如何序列化(json, xml, binary …)？

每秒上万(几十万)次的请求 => 如何保证高吞吐，低延时？

7*24不间服务 => 热布署？采用哪种HA方案？

服务有(无)状态，两次请求是否有上下文关系？

服务有(无)事务，如何解决？

是否方便扩展，是否可以通过简单的加机器来解决性能问题？

……

常见的解决方案

.net => WCF, Remoting
Windows通信基础 (Windows Communication Foundation) 是基于Windows平台下开发和部署服务的软件开发包

Java => EJB, RMI
Enterprise JavaBean sun的服务器端组件模型，最大的用处是部署分布式应用程序

WebService
- 描述数据的方法：XML
- 信息交换的协议：SOAP
- 传输协议：HTTP

RestFull
- HTTP原语封装，回归HTTP本性（get、post、put、delete）
- 业界开放api新标准
- 面向资源开发
- 公开目录结构式的 URI

netflix, dubbo, Hessian, JBoss-Remoting, xxxRPC.....

其他的一些解决方案

facebook => Thrift

Thrift 最初由Facebook于2007年开发，2008年进入Apache开源项目。跨平台通信中thrift可以作为二进制的高性能的通讯中间件，支持数据(对象)序列化和多种类型的RPC服务

google => Protocol buffer

PB是一种用于序列化结构化数据的机制，它具有灵活、高效、自动化的特点。在Google 几乎所有它内部的RPC协议和文件格式都是采用PB

58想做什么？

让一般的程序员能够快速的开发出 和优秀程序员一样高效，安全，稳定的跨平台中间层服务

☺ 学习成本要足够的低，只需要看很少的文档或看个简单的demo就能快速上手

☺ 开发效率要足够高，能够快速的开发出一个高质量的服务

☺ 程序运行速度要足够快，能够轻松应对每秒上万次的QPS，占用机器资源要尽可能的少

☺ 运维要足够的方便，各种指标的监控方便运维和定位问题

☺ 健状性要足够的好，需要有很好的HA和负载均衡机制

☺ 要能够跨平台，58同时有java，c++, .net(少量老项目)项目

怎么做？

服务开发者 => 不需要关心通讯细节

不需要关心序列化细节

不需要关心HA

不需要关心如何跨平台

有很完善的监控和性能计数器功能方便快速定位问题

只专注于具体的业务处理

服务调用者 => 就像调用本地方法一样，其他细节均无需关心

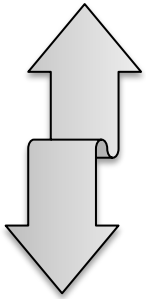
- 通讯组件
- 跨平台序列化组件
- 客户端
- 服务容器


Gaea Client(组件)
负载均衡
动态代理
网络通讯
等待窗口
Gaea Serializer (组件)
Gaea Protocol (组件)
Gaea Server(容器)
网络通讯
过虑器
服务代理
&
热部署
服务监控
权限控制

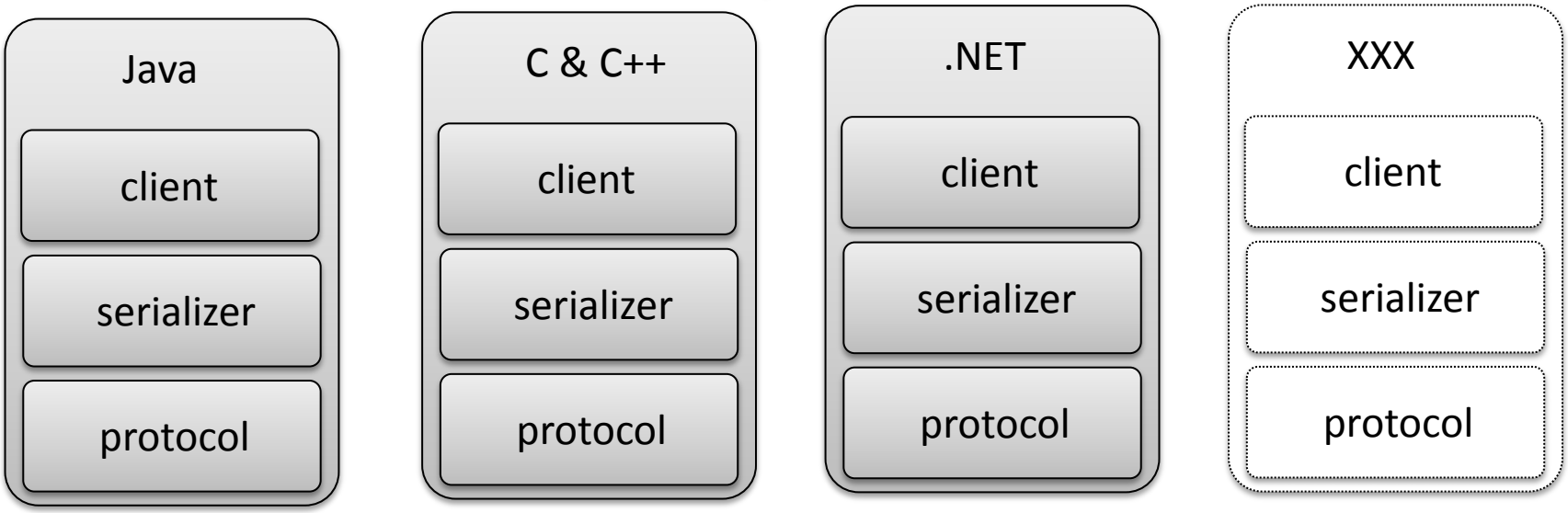
Gaea 实现细节

如何跨平台

服务端：Java开发，宿住在Gaea服务端容器里



TCP长链接，遵守相同的传输协议



Gaea通讯协议

SFP(传输协议)

版本号	1Byte
协义总长度	4Byte
请求编号	4Byte
所采用的序列化类型	1Byte
……(此次略去 xByte)	
所采用的压缩/加密算法	1Byte
消息体(SDP)	nByte

SDP(数据协议)：

- RequestProtocol (请求协议： lookup, methodName, paramKVList)
- ResponseProtocol (响应协议： result, outParam)
- ExceptionProtocol (异常协议：toIP, fromIP, errorCode , errorMsg)

Gaea序列化

. 跨平台 => 定义统一的序列化协议

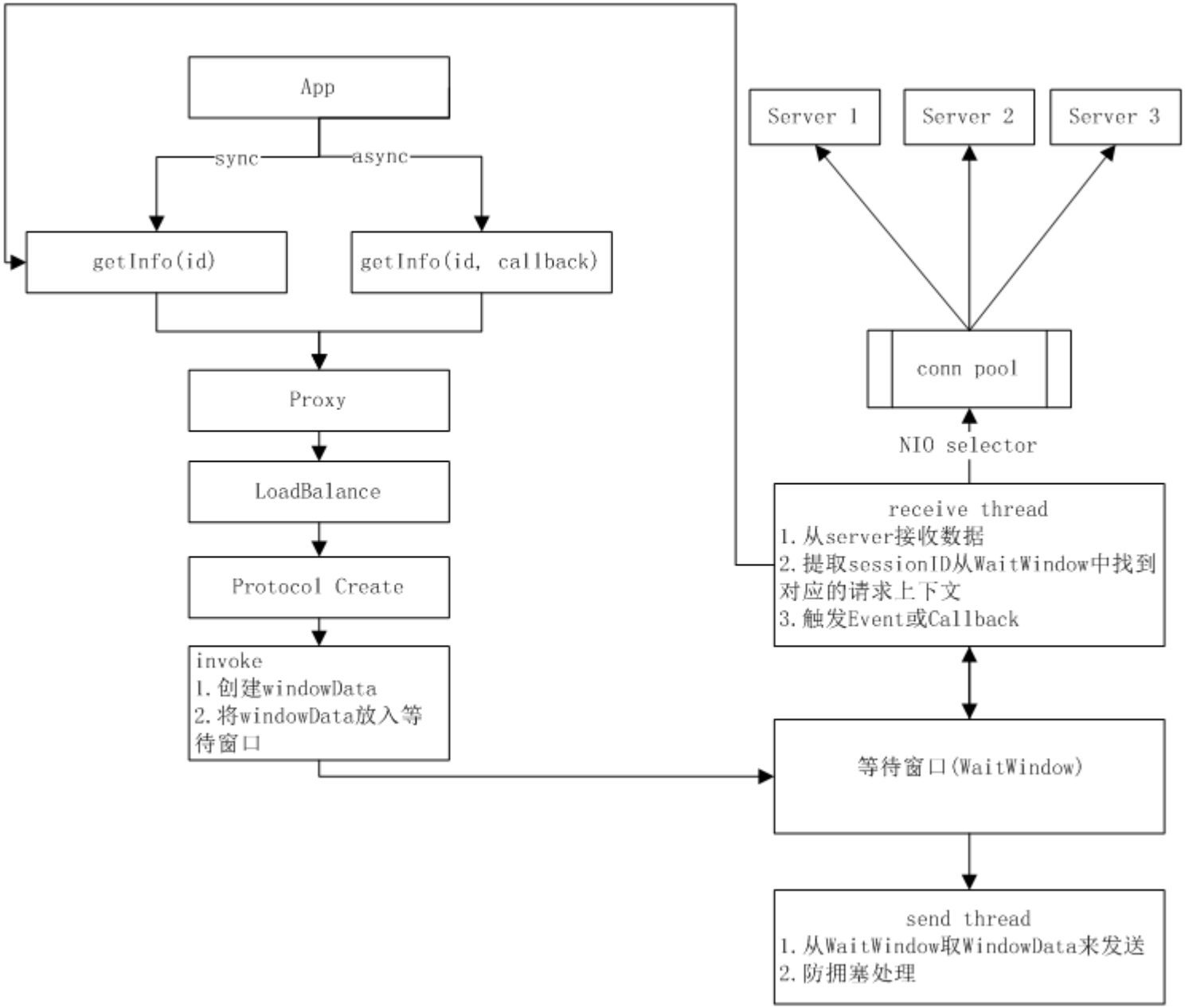
. 序列化出来的数据包要尽可能的小 => 二进制，无元数据

. 支持泛型，支持多重继承

. 使用要尽可能的方便，尽量排除干扰RD的困素，让RD的精力集中在业务处理上(为什么不用pb?)

<table>
<tr><td colspan="7">List</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">1</td><td rowspan="2">4</td><td rowspan="2">4</td><td rowspan="2">4</td><td>类型 Id</td><td>数据</td></tr>
<tr><td>类型 Id</td><td>数据</td></tr>
<tr><td>类型编码</td><td>是否引用</td><td>Hashcode</td><td>数组长度</td><td>泛型参数类型 Id.</td><td colspan="2">类型 Id
数据
数据</td></tr>
</table>

Gaea Client 请求处理模型


App
sync
async
getInfo(id)
getInfo(id, callback)
Proxy
LoadBalance
Protocol Create
invoke
1.创建windowData
2.将windowData放入等待窗口
Server 1
Server 2
Server 3
conn pool
NIO selector
receive thread
1.从server接收数据
2.提取sessionID从WaitWindow中找到对应的请求上下文
3.触发Event或Callback
等待窗口(WaitWindow)
send thread
1.从WaitWindow取WindowData来发送
2.防拥塞处理

动态代理

java.lang.reflect.InvocationHandler

```
@Override
public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    return methodCaller.doMethodCall(args, method);
}
```

```
// tcp://服务名/lookUP
String url = "tcp://comment_demo/CommentService";
CommentService service = ProxyFactory.create(CommentService.class, url);
```

等待窗口(WaitWindow)

send thread
1. 从WaitWindow取WindowData来发送
2. 防拥塞处理

等待窗口(WaitWindow)

wd ...	sessionID event callback data ...	sessionID event callback data ...	sessionID event callback data ...	sessionID event callback data ...	wd ...

receive thread
1. 从server接收数据
2. 提取sessionID从WaitWindow中找到对应的请求上下文
3. 触发Event或Callback

过载保护

一．每一个Server node都有一个请求记数器(请求进来+1 返回后-1)

二．超时时间生成器(保护Client)

a. gaea.config配置的超时时间其实是一个种子时间

b. 超时时间生成器根据种子时间和请求记数器的值动态的生成每次请求的超时时间

三．负载丢弃(保护Server)

HA & 负载均衡

. 一个服务至少同时部署在两台或以上的机器

. Client在对node进行调度的时候可以配置权重

. Client对Server Node健康检测

. Client重试机制

. 无缝重启：

gaea_restart.sh ->
服务端下发重启命令 ->
client收到后之后的请求不再转发给这台server ->
server达到指定的时间后重启 ->
client经过指定的时间周期后偿试链接该server

Server端访问权限控制

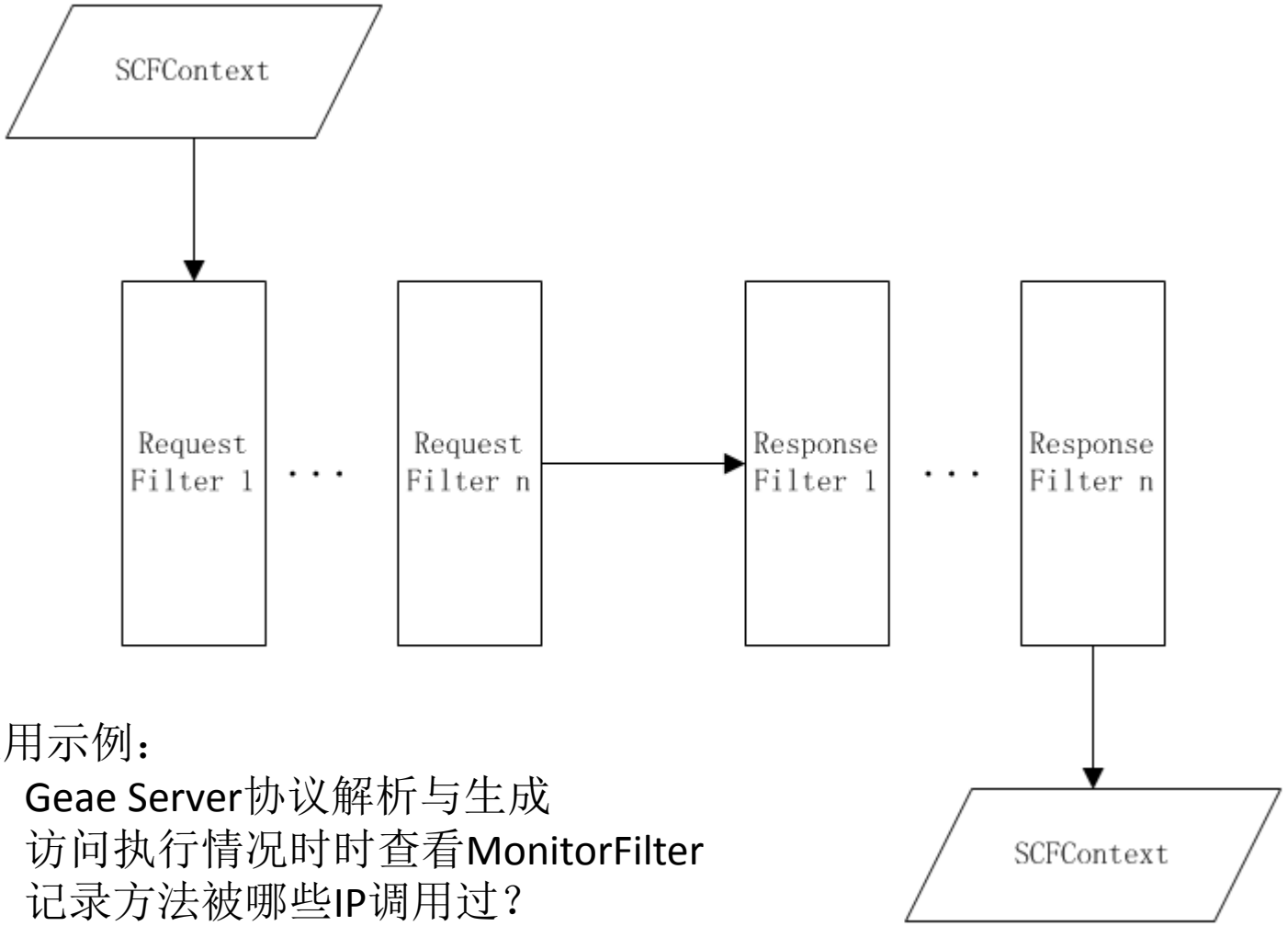
IP黑白名单：

通过IP黑白名单设置(可是指定的ip，也可以是一个网段)达到对访问者进行一个比较粗粒度的控制

方法调用授权：

可以细粒度的进行访问限制(具体过程见下图)

Server Filters



应用示例：

1. Geae Server协议解析与生成
2. 访问执行情况时时查看MonitorFilter
3. 记录方法被哪些IP调用过？

```
count[|second num|method methodName]
        * 查看方法单位时间内的并发数
        * second  : 单位时间 (默认为1s)
        * method  : 方法名
        * example : count
        * example : count|second 3
        * example : count|second 3|method getInfo

time|grep abc[|group num|column -tkda]
        * 查看方法的执行时间
        * grep    : 过虑条件
        * group   : 按指定次数进行汇总
        * column : show column a->all t->time k->key d->description
        * example: time|grep getInfo
        * example: time|grep getInfo|group 10|column -tk

exec|top
    |netstat -na
        * exec command  (at present only allow:top or netstat)
        * example: exec|top

control * 服务相关的控制

help    * 帮助

quit    * 退出
```

服务运行状态时时监控器

telnet

运维

```
bin ----------------------------------> 启动脚本，配制文件
        |------gaea_config.xml -----> Gaea容器配制文件
        |------gaea_log4j.xml -------> 日志配制文件
        |------startup.sh ----------> 启动脚本
        |------startup.bat ----------> 启动脚本
        |------shutdown.sh -------> 关闭脚本
        |------shutdown.bat ------> 关闭脚本
COPYRIGHT ------------> 版权说明
README ---------------> 相关说明
docs -----------------> 文档
log ------------------> 日志文件
lib ------------------> Gaea服务容器自身的和相关依赖的jar
service --------------> 托管服务
        |------deploy ----------> 热布署目录
                |------info_demo （信息管理服务DEMO）
                        |------info.1.0.jar （信息管理服务程序jar）
                                |------xxx.jar ...（信息管理服务依赖的其他jar）
                        |------gaea_config.xml （信息管理服务配置文件）
                        |------gaea_log4j.xml （信息管理服务log4j配置文件）
                |------comment_demo （评论服务DEMO）
                        |------comment.1.0.jar （评论服务程序jar）
                         |------xxx.jar ...（评论服务依赖的其他jar）
                        |------gaea_config.xml （评论服务配置文件）
                        |------geae_log4j.xml （评论服务log4j配置文件）
        |------lib ----------------> 所有服务公用的组件（例如 jdbc驱动, memcache_client）
```

启动：./startup.sh 服务名 例：./startup.sh comment_demo
关闭：./shutdown.sh 服务名 例：./shutdown.sh comment_demo

来看一个基于Gaea开的demo

Just try it~

第一步：定义接口，打上相关的注解

```
6  @ServiceContract  //标记该接口对外公开
7  public interface CommentService {
8
9      @OperationContract  //标记该方法为对外公开的服务方法
10     public Comment getCommentByID(long id) throws Exception;
11
12 }
13
```

第二步：写具体的实现，打上相关的注解

```
5  @ServiceBehavior(lookUP="CommentService")  //标记该类对外提供服务
6  public class CommentServiceImpl implements CommentService {
7
8      @Override
9      public Comment getCommentByID(long id) throws Exception {
10         // TODO Auto-generated method stub
11         return null;
12     }
13
14 }
```

第三步：在传输的实体上打相关的注解

```
 8 @SCFSerializable  //标记该类需要序列化
 9 public class Comment {
10
11     private static final long serialVersionUID = 1L;
12
13     @SCFMember //标记该字段为需要序列化字段
14     private long id;
15
16     @SCFMember
17     private String title;
18
19     @SCFMember
20     private String content;
21
22     @SCFMember
23     private Date addTime;
24
25     //略去getXXX, setXXX等方法
```

Done!!!　服务开发完成

Client端调用示例：

```
9  public class Client {
10
11     @Test
12     public void testGetCommentByID() throws Exception {
13         String url = "tcp://comment_demo/CommentService"; // tcp://服务名/lookUP
14         CommentService service = ProxyFactory.create(CommentService.class, url);
15         Comment comment = service.getCommentByID(123456789L);
16
17         Assert.assertEquals(123456789L, comment.getId());
18     }
19
20 }
```

Done!!!　客户端如同调用本地方法

Gaea的生产应用

58.com基乎所有的服务都是基于Gaea开发的(大大小小服务加起大约有100多个)

其中访问量比较高的几个服务：

信息管理服务 (58.com的核心服务，已经生产环境上运行了快三年)
QPS: 1.4W

用户管理服务
QPS: 8K

安全性要求比较高的服务： 58交易系统的相关服务

后续

Gaea RoadMap

一．写个Eclipse插件集成到Eclipse方便开发人员调试（像tomcat一样）

二．自动生成不同平台接口的功能

三．应用程序域的概念

四．运维监控平台

关于开源

Gaea开源啦～

命名为Gaea(盖亚)

之后58.com SPAT所有开源的项目都以希腊神话中的人物名称来命名项目

后续还会有： WF(web开发框架)

github: https://github.com/58code/Gaea

SACC

Q&A

SPAT

58同城.com

谢谢！